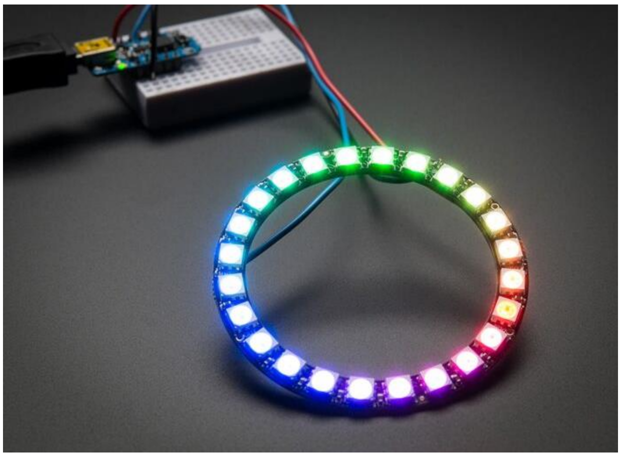
WS2812 LEDs using SPI with STM32

In this tutorial, we will interface the WS2812 LED using SPI with STM32. We will only use the SPI MOSI pin to send the data to the LEDs. So this setup still needs only 1 pin to interface the WS2812 driver to the MCU.



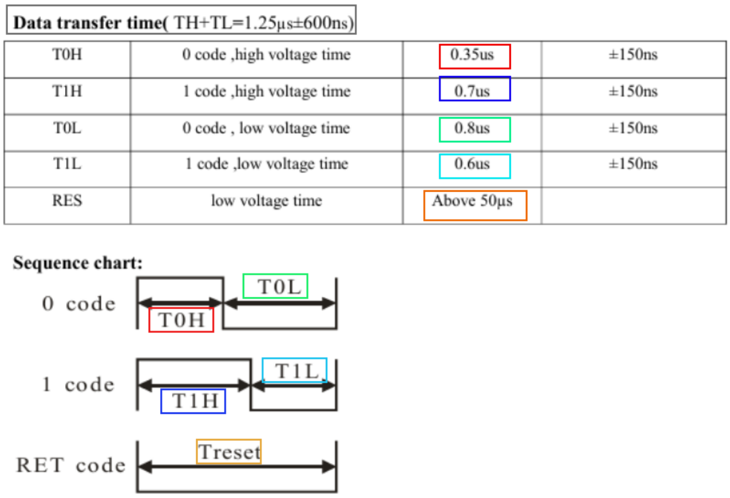
SPI is being used because of its high baud rate. The bit timing required for the LEDs is very low (around 0.4us) and with SPI we can achieve this.

The idea is very simple. If we set the SPI baud rate at 2.5Mbits/s, each bit would represent a pulse of 0.4us.

$$\frac{1}{2.5M/s} = 0.4us$$

According to the datasheet of the WS2812, a bit is considered as 0 if the signal remains HIGH for 0.35us and LOW for 0.8us. And for a bit to be considered as a 1, the signal remains HIGH for 0.7us and LOW for 0.6us. Of course this is flexible with an error up to ±150ns.

Also the reset signal is supplied by pulling the data line low for more than 50us.



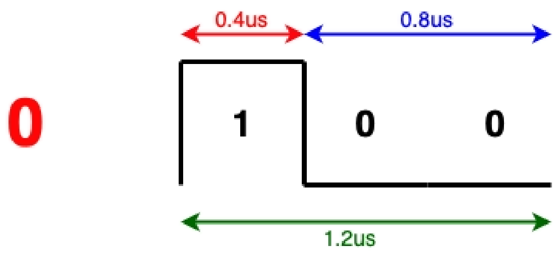
Data transfer time(TH+TL=1.25μs±600ns)

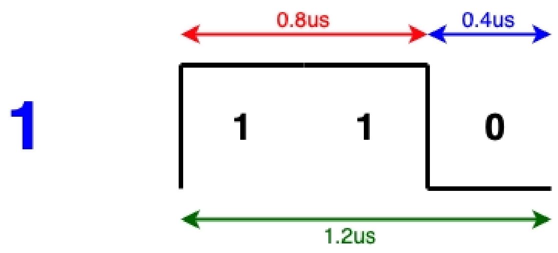
T0H	0 code ,high voltage time	0.35us	±150ns
T1H	1 code ,high voltage time	0.7us	±150ns
T0L	0 code , low voltage time	0.8us	±150ns
T1L	1 code ,low voltage time	0.6us	±150ns
RES	low voltage time	Above 50μs	

Sequence chart:

0 code
T0H
T0L
1 code
T1H
T1L
RET code
Treset

We will use 3 SPI bits to represent a single bit for the WS2812. The 3 SPI bits will have the time period of 1.2us (0.4*3).





As shown above, to send a 0 to the driver, we can send the bits 100. This will keep the pulse high for 0.4us and low for 0.8us. Which is as per the timing required for a 0 in the datasheet.

Similarly to send a 1 to the driver, we can send the bits 110. This will keep the pulse high for 0.8us and low for 0.4us. Which is the acceptable timing for a 1 as per the datasheet.

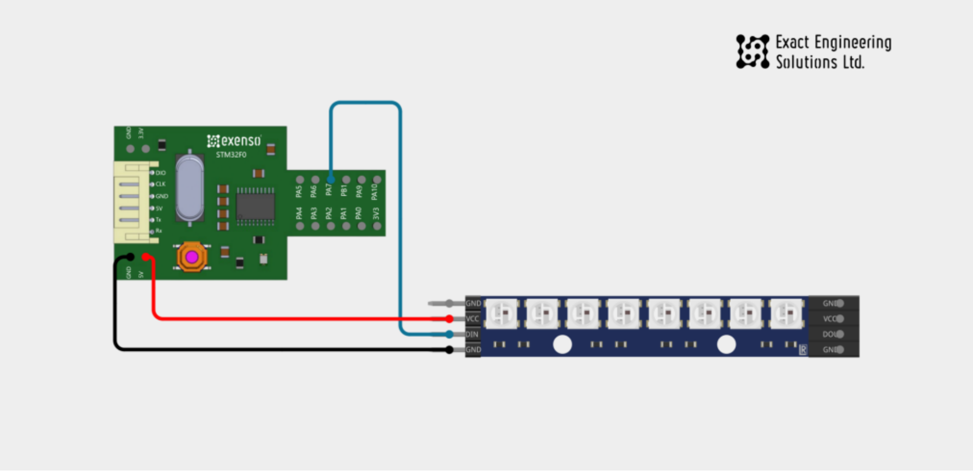
Components Required

You will need the following components –

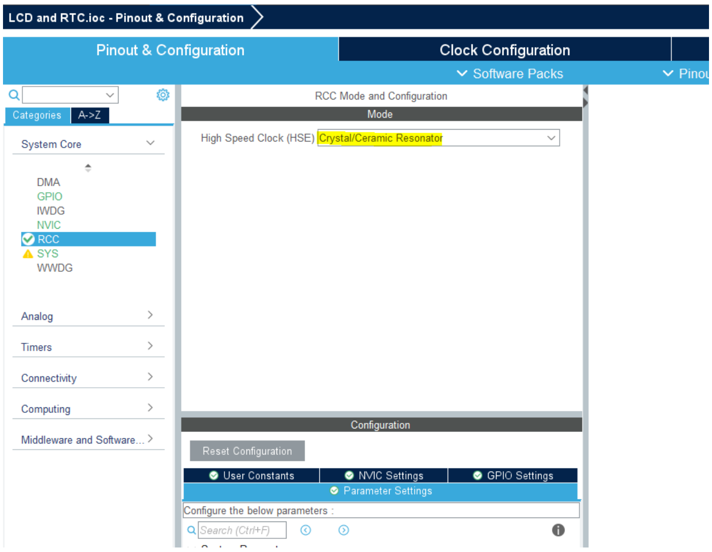
- 1 × Breadboard
- 1 × STM32F030F4P6
- 1x NeoPixel Ring - 24 x WS2812
- Some Jumper wire

Procedure

Follow the circuit diagram shown in the image given below.

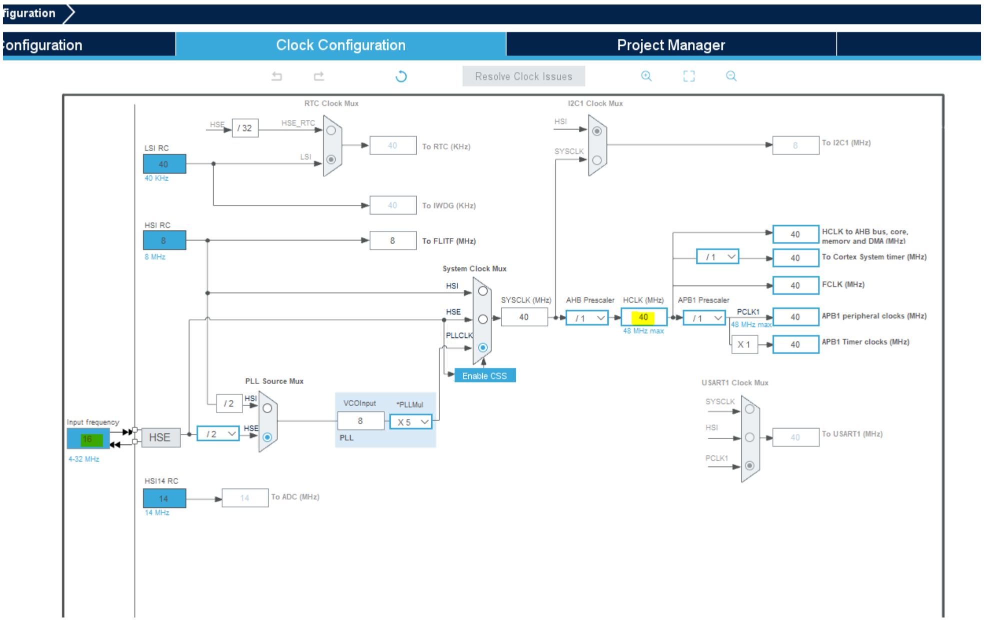


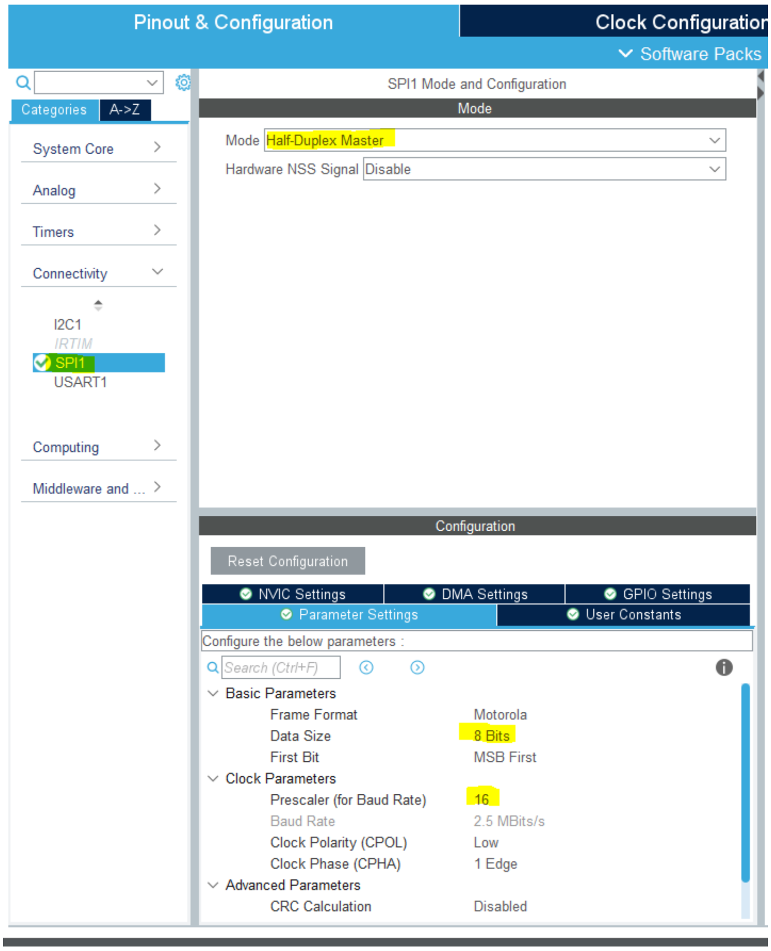
STM32F0 Pin Configuration:



High Speed Clock (HSE) Crystal/Ceramic Resonator

Configure the below parameters :





SPI1 Mode and Configuration

Mode

Mode Half-Duplex Master

Hardware NSS Signal Disable

Configuration

Configure the below parameters :

Basic Parameters

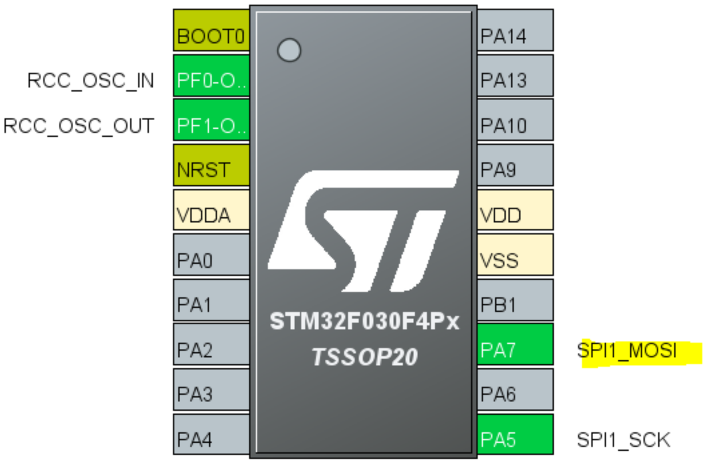
Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First

Clock Parameters

Prescaler (for Baud Rate)	16
Baud Rate	2.5 MBits/s
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	1 Edge

Advanced Parameters

CRC Calculation	Disabled


BOOT0
RCC_OSC_IN
PF0-O..
RCC_OSC_OUT
PF1-O..
NRST
VDDA
PA0
PA1
PA2
PA3
PA4
PA14
PA13
PA10
PA9
VDD
VSS
PB1
PA7
SPI1_MOSI
PA6
PA5
SPI1_SCK
STM32F030F4Px
TSSOP20

Code

```
#include "main.h"
#include "WS2812_SPI.h"
SPI_HandleTypeDef hspi1;

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);

int brightness = 90; //brightness range 0 to 100
int main(void)
{
  HAL_Init();
  SystemClock_Config();

  MX_SPI1_Init();
  for (int i=0; i<24; i++)
  {
        setLED(i, 0, 0, 0);
  }
  WS2812_Send();

  setLED(0, 201, 0, 255);
  setLED(23, 255, 209, 0);
  WS2812_Send();
  HAL_Delay(2000);
  while (1)
  {

               for (int i=0; i<24; i+=1)
               {
                     setLED(i, 255,0,0); // Red color
               }
               WS2812_Send();
               HAL_Delay(1000);

               for (int i=0; i<24; i+=1)
               {
                     setLED(i,0, 255, 0); // Green color
               }
               WS2812_Send();
               HAL_Delay(1000);

               for (int i=0; i<24; i+=1)
               {
                     setLED(i,0,0,255); // Blue color
               }
               WS2812_Send();
               HAL_Delay(1000);

  }

}
```

Output: